TRANSLATING UPDATES OF RELATIONAL DATABASE VIEWS

by

Stavros Stylianos Cosmadakis

B.S., Massachusetts Institute of Technology
(1981)
B.S., Massachusetts Institute of Technology
(1981)

Submitted to the Department of

Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements

for the Degree of

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1983

Signature of Author____.

Department of Electrical Engineering and Computer Science
December 1, 1982

Certified by____.

Christos H. Papadimitriou
Thesis Supervisor

Accepted by____.

A.C. Smith
Chairman, Department Committee

Acknowledgements

I wish to thank my supervisor, Professor Christos Papadimitriou, for pointing out the problem and suggesting the specific methodology to attack it, and also for providing me with key insights at various stages of this research.

I am also grateful to Paris Kanellakis for many helpful discussions and constructive criticism.

This work was supported (in part) by the National Science Foundation under Grant No. MCS 81-20181.

Table of Contents

TRANSLATING UPDATES OF RELATIONAL DATABASE VIEWS

by

Stavros Stylianos Cosmadakis

Submitted to the Department of Electrical Engineering and Computer Science

on December 1, 1982, in partial fullfilment of the requirements

for the Degree of Master of Science in

Computer Science and Engineering

Abstract

We study the problem of translating updates of database views. We disambiguate a view update by requiring that a specified *view complement* (i.e. a second view which contains all the database information omitted from the given view) remains constant during the translation. We study some of the computational problems related to the application of this general methodology in the context of relational databases.

We consider, for the most part, databases consisting of a single relation, with functional dependencies as the only integrity constraints; we also restrict our attention to views defined by projections. We first give a characterization of complementary views (valid also in the presence of join dependencies), which leads to efficient algorithms for checking if two given views are complementary and for determining a non-redundant complement of a given view. We also show that the problem of finding a minimum complement of a given view is *NP*-complete.

We then study in detail the problem of translating the insertion of a tuple into a view. We show how to do the translation in case the insertion is translatable, and we also develop a polynomial time algorithm for testing translatability; we also give two stronger, more efficient translatability tests. We show lower bounds for the complexity of the translatability problem, by proving that it becomes Π_2^P-hard if the view is given in an exponentially succinct way; an

analogous result is shown for one of the stronger tests. We also examine the problem of determining a complement which renders a given insertion translatable; we find that it can be solved in time polynomial in the view, but becomes *NP*-hard if the view is given in an exponentially succinct way; again, analogous results are valid for the stronger tests.

The above results are extended, in a straightforward way, to the cases of deletion and replacement of a tuple. Finally, we define and study a new kind of functional dependencies which is important in the context of complements, the explicit functional dependencies (*EFD*'s), which intuitively state that some part of the database information can be computed from the rest. We examine the interaction of *EFD*'s with functional dependencies and join dependencies, and we also extend our characterization of complementary views to allow for the presence of *EFD*'s.

Thesis Supervisor: Dr. Christos H. Papadimitriou

Title: Associate Professor of Computer Science and Engineering.

Keywords: Relational database, view update, view complement, projective view, functional dependency, polynomial-time hierarchy.

1. Introduction

1.1. The Problem and the Approach

In database systems, the amount and structure of the stored data is decided by the database administrator. However, individual users often want to deal with only part of the information in the database, and moreover they may want to restructure it in a way suitable to their needs. For this reason, database systems often provide the *view* facility. A view is defined by giving a query on the whole database. At any point, the contents of the view is just the outcome of this query. The user queries and updates the view as though it were a database in itself, with no reference to the underlying database. The view idea spares the user from the conceptual complexities of the whole database, makes queries easier by "factoring out" a common subexpression, and can serve as a protection mechanism by restricting access to only insensitive information. A view facility is an important part of many relational database systems, e.g. PRTV [T], QBE [Zl], System R [As] and INGRES [SWKH] (as well as of database systems designed along the lines of the network data model, like DBTG [CO], or the hierarchical data model, like IMS [D, I]).

In relational database systems, a view is in general implemented by naming and storing its definition, which is just a query definition in the query language of the system. Queries on the view are translated into database queries by composing them with the view definition. Thus, querying a view presents no serious conceptual problems.

What is much more complex is the subject of *updating* a view. A simple update operation, such as inserting a tuple in the view, may create formidable problems. The underlying database update may be ambiguous, ill-defined, create inconsistencies in the database, or have side-effects on the view. This problem seems to be related to such fundamental issues as *null values* [Co4, Za2] and *update anomalies* [Col, Co3, BBG] in relational databases. Most existing systems do not allow updates of views (e.g. PRTV, QBE), or allow them only in the trivial case in which the view consists of one of the database relations. This omission apparently reflects our poor understanding of the subject.

In one of the first works dealing with view updates, Dayal and Bernstein [DB] stipulated a notion of *correct translation* of a view update, and gave some straightforward conditions for the existence of such translations . From this and subsequent works, e.g. [RS, Ca, FSD], it became apparent that we need a method for *assigning semantics* to view updates. This method should be *formal* (resolving the delicate ambiguities involved) and *simple* (so the users would define the semantics themselves, perhaps with the aid of the query system).

An excellent solution to this problem was suggested in the work of Bancilhon and Spyratos [BS, Sp]. They developed an elegant theory (quite independent of the relational model) of database mappings, i.e. functions from database states to database states. A view v is such a mapping, and so is an update u on the view. How can we translate u? The translation, T_u, must be such that the updated database maps via v into the updated view. As may be suspected, there are typically many T_u's, so the problem remains. Bancilhon and Spyratos resolve this ambiguity by the notion of the *complement* of a view. A complement of v is another view v', such that the mapping $s \rightarrow (v(s), v'(s))$ (where s denotes the database state) is one-to-one. In other words, any information lost by v can be recovered by v'. A view has many complements (for example, the identity mapping is a complement of *all* views). *Choosing a complement that must remain constant assigns unambiguous semantics to a view update.* The scenario is the following: A user defines a view. Before updating the view, the user must define (probably with the assistance of the system) another view (a complement of the first), which must be held constant during updating (this corresponds to the "rectangle rule" of [Ch] and the "absence of side-effects" of [DB]). Using this information, the system translates (or rejects as untranslatable) the user's updates.

Translating under constant complement amounts to finding a database state s' such that $v(s')=uv(s)$ and $v'(s')=v'(s)$. By the definition of a view complement, s' will be unique if it exists at all. Thus, if such an s' can be found for any s (in which case we say that u is v'-translatable), we can translate u as the database update $T_u=(v \times v')^{-1}(uv \times v')$. The soundness of the overall approach is demonstrated by the following facts [BS]:

i) T_u is *consistent*, i.e. the updated database always maps, under the view definition v, on

the updated view (formally $vT_u = uv$); also T_u is *acceptable*, meaning that if u does not change the view, no change is made on the database either (i.e. for all s, $uv(s)=v(s)$ implies $T_u(s)=s$).

ii) Suppose U is a set of view updates which is reasonable in the sense that it satisfies minimal user requirements, i.e. it is closed under composition and there is a means to cancel the effect of every allowed update on the view (formally, if u, $w \in U$ then $uw \in U$, and if s is a database state and $u \in U$, there is an update $w \in U$ such that $wuv(s)=v(s)$). If v' is a view complement such that any update in U is v'-translatable, then the mapping which associates to an update u in U the database update T_u is a morphism, i.e. $T_{uw}=T_uT_w$ for all u, $w \in U$ (clearly, any reasonable way to translate a set of updates should have this property, i.e. the result of the translation should be the same whether the user applies two updates from the set one after the other or their composite update). On the other hand, the converse also holds: if T is a mapping on U such that, for every $u \in U$, $T(u)$ is a consistent and acceptable database update, and also T is a morphism (i.e. T is a reasonable way to translate view updates into database updates), then there is a view complement v' such that, for every $u \in U$, u is v'-translatable and $T(u)=T_u$.

However, as was pointed out earlier, this approach is essentially independent of any particular data model. In this work we investigate some of the issues and problems which arise when one attempts to apply this methodology in the context of the *relational model*, with a view towards rendering it realizable in practice. In the remaining part of the Introduction we review briefly some basic concepts and notations of the relational model, and we also give an overview of the results obtained.

1.2. The Relational Model - Basic Definitions

The relational model [Col] assumes that the data are stored in two-dimensional tables called *relations*. The columns of a table correspond to *attributes*, and the rows to *tuples* (*records*). Each attribute A has an associated *domain* of values D_A, and a tuple is viewed as a mapping from the attributes to their domains. We use the letters A, B, C, ... to denote attributes and the letters ... ,

X, Y, Z to denote sets of attributes. A *relation scheme* is a finite set of attributes labeling the columns of a table, and is usually written as a string of attributes. If X is a relation scheme labeling the columns of a relation R, we say that *R is defined over X*. A *database scheme D* is a finite set of relation schemes. A *database over D* is a set of relations containing exactly one relation over each relation scheme in D.

In the context of the relational model, one way of formulating queries (and thus of defining views also) is by using a set of operators defined on relations (relational algebra [Col, Co2]). The operators that will be of primary interest to us are *projection* and *join*. The projection of a tuple t to a set of attributes X, written $t[X]$, is simply the restriction of t to X; the projection of a relation R to X, written $\pi_X(R)$, is the set of projections of the tuples in R to X. If R_1, R_2 are relations defined over the relation schemes X_1, X_2 respectively, the join of R_1 and R_2, written $R_1 * R_2$, is the relation over $X_1 \cup X_2$ consisting of all the tuples μ such that $\mu[X_1] \in R_1$, $\mu[X_2] \in R_2$.

Semantic information is captured by means of *integrity constraints* (i.e. predicates on relations), usually expressed as first-order sentences called *dependencies* [Col]. Various kinds of dependencies have been defined and studied in the literature; we will be primarily concerned with *functional dependencies* and *multivalued dependencies* (the latter are a special case of *join dependencies*).

A *functional dependency* (*FD*) [Ar, Col] is a statement of the form $X \rightarrow Y$, where both X and Y are sets of attributes. The *FD $X \rightarrow Y$ holds* in a relation R if for all tuples μ and ν of R, if $\mu[X] = \nu[X]$, then $\mu[Y] = \nu[Y]$.

A *join dependency* (*JD*) [R2] is a statement of the form $*[R_1, \ldots, R_q]$, where each R_i is a relation scheme. The *JD $*[R_1, \ldots, R_q]$ holds* in a relation R if $* \, \pi_{R_i}(R) = R$. A *multivalued dependency* (*MVD*) [F, Za1] is a *JD* with at most two relation schemes. An *MVD* $*[R_1, R_2]$ is also written as $R_1 \cap R_2 \rightarrow\rightarrow R_1$ (or equivalently $R_1 \cap R_2 \rightarrow\rightarrow R_2$).

Finite sets of dependencies will be denoted by Σ. A *database schema S* is a pair (D, Σ), where D is a database scheme. An *instance* (i.e. a database DB over D) which satisfies the dependencies in Σ is called *legal* (notation: $DB \models \Sigma$).

For a more detailed exposition of the fundamendal notions and notations of the relational model and of the relevant theory, see [U1].

1.3. Outline of the Thesis

This work is mainly centered around the study of the computational problems arising when one attempts to apply the general methodology proposed by Bancilhon and Spyratos in the context of the relational model. We discover that very interesting theoretical questions already arise at very simple cases of the application. In particular, we concentrate on database schemas consisting of *a single relation*, with integrity constraints which are (for the most part) just *functional dependencies.* The views we consider are simply *projections* of the relation. Working with a single relation corresponds to some unrealistic *universal relation assumption* [U2], but it yields a simplified problem which must be conquered first. Functional dependencies constitute a simple and practical class of constraints. Projective views are, again, the simplest imaginable, and they are also important from a practical point of view.

In Section 2 we characterize when two projections are complements of each other. There is an interesting parallel between this characterization and the notion of *independence* of Rissanen [R1]. Our necessary and sufficient condition (which can be generalized to include the presence of *join dependencies*) states that the common part of the projections must be a *superkey* of one of the projections. As a consequence, it is easy to test whether two given projections are complementary in a schema. It is also possible to construct a nonredundant (minimal) complement of a given projection in polynomial time. Unfortunately, finding a *smallest* (i.e. with fewest attributes) complement of a given projection is shown to be *NP*-complete.

In Section 3 we study how to implement the insertion of a tuple into a projection, keeping a given complementary projection unchanged. We show that this can be done in a unique way, and so the problem reduces to testing whether the resulting database is consistent. We show that this test can be carried out in time cubic *in the number of tuples of the view.* Since this is likely to be impractical, we also develop two alternative *stronger* tests that can be executed more

efficiently.

Ideally, we would like the time complexity of our update algorithms to depend on the number of attributes, functional dependencies, and other parameters of the *schema*, not of the instance. When the time must depend on the number of tuples, we would at least like this dependence to be *logarithmic*, since this number is expected to be very large. However, complexities like those described in the previous paragraph resemble, in a practical sense, *exponential* complexities. We show some negative complexity results which suggest that this "exponential" behavior is inherent: The translatability problem becomes Π_2^P-hard [St] if the view is represented in some exponentially succinct way (e.g., as the union of two Cartesian products). Even one of the simpler, stronger tests mentioned above becomes co-*NP*-hard.

Finally, we examine the complexity of finding a complement which renders a given insertion translatable. We show that this problem is polynomial in the number of tuples of the view, but inherently exponential in the size of the schema (and the *logarithm* of the number of tuples of the view). Similar results can be obtained for the two stronger tests.

In Section 4 we extend these results to the case of *deletions* and *replacements* of tuples. We find that, for the most part, the extension is rather straightforward. Finally, in Section 5 we define and examine a new kind of functional dependencies which is important in the context of complements, the *explicit functional dependencies*. We extend our characterization of complementary projections to also allow for the presence of explicit functional dependencies. Section 6 concludes this work by pointing out some directions for further research.

2. Defining a Complement

Let S be a database schema (U, Σ), where U is a universal set of attributes and Σ is a finite set of dependencies. A relation R over U (*instance* of U) is called *legal* if it satisfies all the dependencies in Σ (notation: $R \models \Sigma$). A *view* of S is for us a projection defined by a subset X of U. For each instance R, the corresponding instance of the view is $\pi_X(R)$. We disambiguate updates on a view by defining a *second view*, Y, the *complement* of X. Two views X and Y are called *complementary* if $\pi_X(R)=\pi_X(R')$ and $\pi_Y(R)=\pi_Y(R')$ imply $R=R'$, whenever R and R' are both legal instances. In other words, the two views together contain enough information to reconstruct the whole database.

When are two views X and Y complementary? Clearly, a sufficient condition is that the *MVD* $*[X, Y]$ holds in every legal instance, i.e. Σ *implies* the *MVD* $*[X, Y]$. If this is the case, the database can be reconstructed from its projections on X and Y by join. Recently it has been shown [V1] that the condition is not necessary, i.e. if Σ consists of general first-order sentences then π_X and π_Y can be complementary without the reconstruction operator being the join. However, we show that this cannot happen if we impose more restrictions on Σ:

Theorem 1:

Let Σ consist of functional dependencies and join dependencies. Then X, Y are complementary iff $\Sigma \models *[X, Y]$.

Proof: The "if" direction is immediate: if Σ implies the *MVD* $*[X, Y]$, then for every legal instance R we have $\pi_X(R) * \pi_Y(R)=R$. Consequently, if for two legal instances R, R' we have $\pi_X(R)=\pi_X(R')$ and $\pi_Y(R)=\pi_Y(R')$, we get $\pi_X(R) * \pi_Y(R)=\pi_X(R') * \pi_Y(R')$ and from this $R=R'$, i.e. X, Y are complementary.

For the "only if" direction, assume that Σ *does not imply* the *MVD* $*[X, Y]$; we will show that X, Y are not complementary, by exhibiting two *distinct* legal instances R, R' for which $\pi_X(R)=\pi_X(R')$ and $\pi_Y(R)=\pi_Y(R')$.

Let σ be a join dependency $*[R_1, \dots, R_q]$; define $M(\sigma)$ to be the set of MVD's $\{ *[\bigcup_{i \in S_1} R_i, \bigcup_{i \in S_2} R_i], S_1, S_2$ a partition of $\{1, \dots, q\} \}$ (see also [MSY]). If Σ' is the set we obtain if we replace each join dependency σ in Σ by the multivalued dependencies in $M(\sigma)$, then, since σ implies each MVD in $M(\sigma)$, Σ implies Σ'; but by our hypothesis Σ does not imply $*[X, Y]$, so Σ' does not imply $*[X, Y]$ either. Now since Σ' consists of FD's and MVD's only, there is a two-tuple counterexample to this implication [SDSF], i.e. there is a relation R consisting of two tuples μ and ν which satisfies all the dependencies in Σ' but does not satisfy $*[X, Y]$.

From the relation R construct another relation R' as follows: since R does not satisfy $*[X, Y]$, it must be that $\mu[X \cap Y] = \nu[X \cap Y]$, and also $\mu[Y-X] \neq \nu[Y-X]$ and $\mu[X-Y] \neq \nu[X-Y]$. Let R' consist of a tuple μ' which agrees with μ on X and with ν on Y-X, and of a tuple ν' which agrees with ν on X and with μ on Y-X. Clearly, $R \neq R'$, R' satisfies all the dependencies in Σ' (it defines the same "*special truth assignment*" [SDSF] as R), and also $\pi_X(R) = \pi_X(R')$ and $\pi_Y(R) = \pi_Y(R')$. Thus, we only need to show that R and R' are both legal, i.e. they both satisfy all the JD's in Σ (they obviously satisfy the FD's in Σ, since these are included in Σ' and R, R' satisfy Σ').

Let $*[R_1, \dots, R_q]$ be a JD in Σ; to show that it holds in R, it suffices to show that, if a tuple ξ is obtained by joining $\xi_1[R_1], \dots, \xi_q[R_q]$, where $\xi_1, \dots, \xi_q$ are tuples of R, then either $\xi = \mu$ or $\xi = \nu$. This is certainly true if $\xi_1 = \dots = \xi_q = \mu$ or if $\xi_1 = \dots = \xi_q = \nu$; else, let $S_1 = \{i: \xi_i = \mu\}$, $S_2 = \{i: \xi_i = \nu\}$. Since the MVD $*[\bigcup_{i \in S_1} R_i, \bigcup_{i \in S_2} R_i]$ is in Σ', it holds in R, and thus either $\xi = \mu$ or $\xi = \nu$. Thus R satisfies all dependencies in Σ, and so does R' (by the same argument). This completes the proof. ∎

Notice that our condition (though not the proof) parallels the result of Rissanen on *independence* [R1]. Intuitively, independence is stronger than complementarity, and thus our Theorem contains only the first condition of [R1]. To see why, consider the classical Employee-Department-Manager scheme. The decomposition into $X = ED$, $Y = EM$ is not independent, although X and Y are complementary.

Theorem 1 has some algorithmic consequences:

Corollary 1:

Given (U, Σ), X, $Y \subseteq U$, whether X, Y are complementary can be tested in polynomial time.

Prrof: By Theorem 1, testing for complementarity amounts to inferring an *MVD* from a set of *FD*'s and *JD*'s. The latter can be done in polynomial time [MSY, V2]. ▮

Corollary 2:

Given (U, Σ) and $X \subseteq U$, we can find in polynomial time a minimal (nonredundant) complement of X.

Proof: Simply start with the trivial complement U and repeatedly take out any attribute in X which can be taken out without affecting complementarity (examine the attributes in some arbitrary order). ▮

Thus we can program in a database system some guidance to the user towards the definition of a complement. Unfortunately, as so often happens, finding the minimum is much harder.

Theorem 2:

Given (U, Σ), $X \subseteq U$ and $k > 0$, determining whether there is a complement Y of X with $|Y| = k$ is *NP*-complete.

Proof: Membership in *NP* is obvious: just guess a subset Y of U with $|Y| = k$ and verify (Corollary 1) that X, Y are complementary.

To prove the hardness part, we will make a reduction from the 3-satisfiability problem (3-SAT), which is known to be *NP*-complete [Ck, K, GJ]. Let φ be a Boolean formula in 3-conjunctive normal form (3-CNF); let x_i, $i = 1,...,n$, be the variables occuring in φ, and let f_j, $j = 1,...,m$, be the clauses of φ. We construct the following schema $S_\varphi = (U, \Sigma)$: U is

$F_1...F_mX_1X'_1...X_nX'_nA$ and Σ contains the functional dependencies $F_1...F_mX_i \rightarrow X'_i$, $F_1...F_mX'_i \rightarrow X_i$, $i=1,...,n$, and also for each clause $f_j = l_{j1} + l_{j2} + l_{j3}$, $j=1,...,m$, the functional dependencies $L_{j1} \rightarrow F_j$, $L_{j2} \rightarrow F_j$, $L_{j3} \rightarrow F_j$ (if $l_{ji} = x_r$, $L_{ji} = X_r$; if $l_{ji} = \neg x_r$, $L_{ji} = X'_r$).

Now let X be $F_1...F_mX_1X'_1...X_nX'_n$; we claim that X has a complement Y with $|Y| = 1+n$ iff φ is satisfiable. To see this, first assume that φ is satisfiable, and let h be a satisfying assignment. Take Y to be $L_1...L_nA$, where $L_i = X_i$ if $h(x_i)$ is true, $L_i = X'_i$ if $h(x_i)$ is false. To show that X, Y are complementary, it suffices to show (by Theorem 1) that $\Sigma \models {*}[X, Y]$; to do that, we use the *chase* method for inferring dependencies [MMS]: if we consider the *tableau* consisting of a row with distinguished variables in the X columns and a row with distinguished variables in the Y columns, then we can convert the second row into a row of distinguished variables by using the *FD*-rules corresponding to the *FD*'s in Σ as follows: first, since h satisfies f_j, at least one of the *FD*'s $\{L_{j1} \rightarrow F_j, L_{j2} \rightarrow F_j, L_{j3} \rightarrow F_j\}$ can be used to fill in F_j, and this can be done for all j. Then the *FD*'s $F_1...F_mX_i \rightarrow X'_i$, $F_1...F_mX'_i \rightarrow X_i$ can be used to fill in the remaining X_i's and X'_i's.

For the converse, suppose there is a complement Y of X with $|Y| = 1+n$. Clearly Y has to contain at least one of $\{X_i, X'_i\}$ (else there is no way to fill in both X_i and X'_i), and thus Y contains exactly one of $\{X_i, X'_i\}$ for each i (also $A \in Y$). Consider now the assignment h, where $h(x_i)$ is true if $X_i \in Y$ and false if $X'_i \in Y$: since F_j is filled in, at least one of $\{L_{j1}, L_{j2}, L_{j3}\}$ must be contained in Y, and thus h satisfies f_j. This is true for all j, so h satisfies φ and the claim is established.

Finally, it is easy to see that S_φ and X can be constructed in time polynomial in the length of φ. This completes the proof. ∎

Observe that in our reduction we only used *FD*'s, so Theorem 2 is true even if Σ is constrained to contain only *FD*'s. Now if $\Sigma' = \{Z \rightarrow\rightarrow B \mid Z \rightarrow B$ is an *FD* in $\Sigma\}$, then if σ is a *JD* $\Sigma \models \sigma$ iff $\Sigma' \models \sigma$ [BV]. Thus, we might as well replace Σ by Σ' in our proof, which means that Theorem 2 is true even if Σ is constrained to consist of *MVD*'s only.

3. The Translation of Insertions

3.1. Testing Translatability

Σ is now a set of functional dependencies: we furthermore assume that each FD in Σ is of the form $X \to A$, where A is a single attribute (this is easy to enforce, by replacing each FD $X \to Y$ in Σ by the equivalent set of FD's $\{X \to A: A \in Y\}$).

Suppose that the view X and its complement Y are given, and so is the current instance V of the view. We wish to translate the update u on the view consisting of the *insertion* of a tuple t, while keeping the complement $\pi_Y(R)$ constant. How can we design an update on R, T_u, which achieves this?

The translation T_u should have certain obvious properties:

A. It should implement the view update, that is $\pi_X(T_u[R]) = V \cup t$.

B. It should keep the complement constant, according to the prescribed semantics; that is, $\pi_Y(T_u[R]) = \pi_Y(R)$.

C. It should yield a consistent database, that is, if R is a "possible" instance, $T_u[R] \models \Sigma$. The meaning of "possible" is the subject of property D below.

D. A more subtle but important assumption is that the user proposes the update *based on his knowledge of the view* and on no other information concerning the database. Thus, the translation should produce a legal database *for all legal instances* of the overall database, given the instance of the view.

It is quite interesting that these properties determine precisely when the insertion of a tuple t in an instance V of the view is translatable, and, if it is, the translation T_u is unique.

First, suppose that $t \notin V$ (otherwise T_u is the identity). Since $\pi_Y(R)$ must be kept constant (Property B) we must assume that $t[X \cap Y] \in \pi_{X \cap Y}(R) = \pi_{X \cap Y}(V)$; otherwise, the only way to insert t in $\pi_X(R)$ (Property A) would be to insert something in $\pi_Y(R)$. By Theorem 1, $X \cap Y$ is a superkey of either X or Y. If it is a superkey of X, then the update is clearly untranslatable, because $V \cup t$ is not the projection of a legal instance (Property C). So $X \cap Y \to Y$. It follows that

the only T_u satisfying A, B and C is the insertion of the *tuple* $t^*\pi_Y(R)$ in the database R: $T_u[R]=R \cup t^*\pi_Y(R)$ (* denotes the natural join).

It remains to determine under which conditions $T_u[R]$ is legal (Property C). The insertion of t into V is translatable iff $T_u[R]\models\Sigma$ *for all* R *such that* $R\models\Sigma$, $\pi_X(R)=V$ (Property D was used here).

Suppose that the insertion is not translatable. This means that there is a functional dependency, say $Z\to A$, which is violated by $T_u[R]$ for some R for which $R\models\Sigma$ and $\pi_X(R)=V$. Since R satisfies $Z\to A$, the inserted tuple must be the culprit. Thus, there must be a tuple r of V which agrees with t on $Z\cap X$ and, if $A\in X$, disagrees with t on A. Furthermore, if we fill the rows of V with new symbols in the columns of Y-X, only with $r[Z\cap(Y\text{-}X)]=\mu[Z\cap(Y\text{-}X)]$ where μ is a tuple agreeing with t on $X\cap Y$ (call this relation $R(V, t, r, Z\to A)$) and then perform the *chase* [MMS] wrt Σ on this relation, no two distinct elements of V, neither the elements corresponding to $r[A]$, $\mu[A]$ (if $A\in Y$-X), are ever equated (if they are, we say the chase *succeeded*). It turns out that this is a necessary and sufficient condition for untranslatability:

Theorem 3:

The insertion of t into V ($t\notin V$) is translatable as $R\leftarrow R \cup t^*\pi_Y(R)$ if and only if

(a) $t[X\cap Y]\in\pi_{X\cap Y}(V)$.

(b) Σ implies $X\cap Y\to Y$, and Σ does not imply $X\cap Y\to X$.

(c) $\text{Chase}_\Sigma[R(V, t, r, f)]$ succeeds for all functional dependencies $f\in\Sigma$ and tuples r of V.

Proof: By the preceding discussion, all we need to notice is that, if $\text{Chase}_\Sigma[R(V, t, r, f)]$ does not succeed for some *FD* $f\in\Sigma$ and some tuple r of V, then it actually provides us with a *counterexample*, i.e. it constructs a relation R such that $R\models\Sigma$, $\pi_X(R)=V$, and $T_u[R]$ violates f. In the opposite case, the chase actually provides us with a *proof* that there can be no relation R such that $R\models\Sigma$, $\pi_X(R)=V$, and $T_u[R]$ violates some $f\in\Sigma$, i.e. $T_u[R]\models\Sigma$ for all R such that $R\models\Sigma$, $\pi_X(R)=V$. ∎

Corollary:

Whether an insertion is translatable can be tested in time $O(|V|^3 log|V| \ |\Sigma|^2 \ |Y\text{-}X|)$.

Proof: Clearly, condition (a) can be tested in time $O(|V|)$, and condition (b) can be tested in time $O(|\Sigma|)$ (using the linear-time algorithm [BB] for inferring an *FD* from a set of *FD*'s). Since condition (c) can be tested by doing $O(|\Sigma| \, |V|)$ chases, it suffices to show that the chase of $R(V, t, r, f)$ can be computed in time $O(|V|^2 log|V| \ |\Sigma| \ |Y\text{-}X|)$. Recall that the chase procedure consists in repeatedly locating a pair of tuples μ, ν such that $\mu[Z]=\nu[Z]$ and $\mu[A]\neq\nu[A]$ for some *FD* $Z\rightarrow A$ in Σ, and replacing the element $\mu[A]$ with $\nu[A]$ throughout the A column. This can be done by the following straightforward algorithm:

Initialize R^* to be $R(V, t, r, f)$.
Repeat until no new change is made on R^*:
  For each *FD* $Z\rightarrow A$ in Σ do:
    Sort R^* lexicographically according to the elements of the Z columns.
    Find the first pair of consecutive tuples μ, ν such that $\mu[Z]=\nu[Z]$, $\mu[A]\neq\nu[A]$.
    Replace $\mu[A]$ by $\nu[A]$ throughout the A column.

It is clear that each execution of the body of the for loop takes time $O(|V| log|V|)$, so each execution of the for loop takes time $O(|V| log|V| \ |\Sigma|)$. Since each time the for loop is executed the number of distinct symbols in the $Y\text{-}X$ columns is reduced by at least one (if the chase ever attempts to equate two different elements in one of the X columns we stop immediately), and initially we have $|Y\text{-}X||V|$ such symbols, the for loop will be executed at most $|Y\text{-}X||V|$ times, and so the total time is at most $O(|V|^2 log|V| \ |\Sigma| \ |Y\text{-}X|)$. ∎

The algorithm described above can be speeded up by taking the following straightforward shortcut: to construct $R(V, t, r, Z\rightarrow A)$, first fill the rows of V with new symbols in the columns of $Y\text{-}X$, *then do a chase* (and store the resulting relation to be re-used for other members of Σ), and then set $r[Z\cap(Y\text{-}X)]=\mu[Z\cap(Y\text{-}X)]$, for some μ agreeing with t on $X\cap Y$. However, since we are still unable to provide a better quarantee for its worst-case performance than $O(|V|^3 log|V|)$,

its applicability in practice is dubious, in view of the fact that $|V|$ is normally very large. For this reason, we will now present two alternative tests for which we can show better upper bounds to their worst-case performance. However, our tests will be *stronger* than necessary, i.e. in addition to rejecting all untranslatable insertions, they may also reject some translatable ones.

Test 1

Our first alternative test consists in simply avoiding to do a full chase on $R(V, t, r, Z\to A)$; instead, for each tuple μ agreeing with t on $X\cap Y$, we do a chase on the two-tuple relation consisting of r and μ, and we report success if any of these chases equates $r[A]$, $\mu[A]$ (if $A\in Y\text{-}X$; notice that in this case $\mu[A]=t[A]$, since $X\cap Y\to Y$), or attempts to equate two distinct elements of V. Thus, what we are actually doing is imposing the extra requirement that $\text{Chase}_{\Sigma}[R(V, t, r, f)]$ *succeeds fast*, if it succeeds at all. Intuitively, this does not seem to be very restrictive, and one may hope that Test 1 will actually accept most of the translatable insertions that will occur in practice.

The test can obviously be implemented in time $O(|V|^2\,|\Sigma|)$. However, we can do better (in terms of the dependence of the time complexity on $|V|$), as follows:

1. Fill the rows of V with new symbols in the $Y\text{-}X$ columns. Then determine the set of tuples $T=\{\mu\colon\ \mu[X\cap Y]=t[X\cap Y]\}$. This can be done in time $O(|V|)$.
2. For each $Z\subseteq U$, construct a copy of the relation T (call it T_Z), and sort it according to the contents of the Z columns. This can be done in time $O(2^{|U|}\,|V|log|V|)$.
3. For each $Z\subseteq U$, compute the *closure* of Z under Σ, i.e. the set $Z^{+}=\{A\colon\ \Sigma\models Z\to A\}$. This can be done in time $O(2^{|U|}\,|\Sigma|)$ (using the algorithm of [BB] for computing closures).
4. For each $Z\subseteq U$, go through the table T_Z from top to bottom and, whenever a tuple agrees with the previous one on Z, make it agree on Z^{+}. This has the effect of making all tuples which agree on Z to agree on Z^{+} (as they should), and it can be done in time $O(2^{|U|}\,|V|)$.

5. For each FD $Z \rightarrow A$ in Σ do:

   For each tuple r for which $r[Z \cap X] = t[Z \cap X]$ and $r[A] \neq t[A]$ (if $A \in X$), do:

   Make r agree with μ on $Z \cap (Y\text{-}X)$, where μ is a tuple in T.

   For each $Z \subseteq U$ do:

   Insert r in T_Z.

   If $r[Z] = \nu[Z]$, where ν is either the tuple next to r or the tuple before r in T_Z, then make r agree with ν on Z^+.

This can be done in time $O(|\Sigma|\ |V|\ 2^{|U|}\ \log|V|)$.

Thus, the overall time expended is $O(|V| log|V|\ 2^{|U|}\ |\Sigma|)$. Of course, there are various optimizations and shortcuts one may employ in an actual implementation (for example, to handle the potential problem of having too many sorted tables - say by actually having for each Z a *sequence of pointers* to the tuples of T). Observe that the running time of this algorithm will be better than our worst-case upper bound for the exact translatability test (and also better than the obvious $O(|V|^2\ |\Sigma|)$ algorithm) if $|V|/log|V| > 2^{|U|}$, which is definitely going to be the case in practical situations.

Test 2

Our second alternative test has a somewhat different flavor: notice that Test 1 saves time by doing only part of the computation necessary for each particular chase. Test 2, instead, will only do one full chase, if this is possible.

More specifically, recall that the essential part of the translatability test (in terms of time requirements) is checking if for all R such that $R \models \Sigma$, $\pi_X(R) = V$, we have $T_u[R] \models \Sigma$. Suppose now that Y actually has the following property:

For all R_1, R_2 such that $R_1 \models \Sigma$, $R_2 \models \Sigma$, $\pi_X(R_1) = \pi_X(R_2)$, $t[X \cap Y] \in \pi_{X \cap Y}(R_1) = \pi_{X \cap Y}(R_2)$, we have that $T_u[R_1] \models \Sigma$ iff $T_u[R_2] \models \Sigma$.

We call such a Y a *good* complement of X. Our interest in good complements lies in the

fact that, if Y happens to be a good complement of X, then clearly all we need to do to test if the insertion u is translatable is consruct some relation R_0 such that $R_0 \models \Sigma$, $\pi_X(R_0) = V$, and test if $T_u[R_0] \models \Sigma$. We can construct such an R_0 by filling the rows of V with new symbols in the Y-X columns and then doing a chase; this can be done in time $O(|V|^2 log|V|\ |\Sigma|\ |Y-X|)$. Testing if $T_u[R_0] = R_0\ \cup\ t^*\pi_Y(R_0)$ satisfies Σ amounts to testing if for each tuple μ of R_0, the two-tuple relation consisting of μ and $t^*\pi_Y(R_0)$ satisfies all the FD's in Σ; this can be done in time $O(|V|\ |\Sigma|)$.

Thus, all we need to do is show how one can test if a given complement Y of X is actually a good complement. Observe that this property is independent of the tuple t to be inserted, i.e. it is a property of the schema only (X, Y and Σ).

Suppose, then, that Y is not a good complement of X. This means that there are two relations R_1, R_2 such that $R_1 \models \Sigma$, $R_2 \models \Sigma$, $\pi_X(R_1) = \pi_X(R_2)$, $t[X \cap Y] \in \pi_{X \cap Y}(R_1) = \pi_{X \cap Y}(R_2)$, $T_u[R_2] \models \Sigma$ and $T_u[R_1] = R_1\ \cup\ t^*\pi_Y(R_1)$ violates some FD in Σ, say $Z \rightarrow A$. Since $R_1 \models \Sigma$, there must be a tuple μ_1 in R_1 such that $\mu_1[Z] = \pi_Z[t^*\pi_Y(R_1)]$, $\mu_1[A] \neq \pi_A[t^*\pi_Y(R_1)]$. Also there must be a tuple ν_1 in R_1 such that $\nu_1[X \cap Y] = t[X \cap Y]$.

Since $\pi_X(R_1) = \pi_X(R_2)$, we can then find two tuples μ_2, ν_2 of R_2 such that $\mu_2[X] = \mu_1[X]$, $\nu_2[X] = \nu_1[X]$. Now consider the relations R'_1, R'_2, consisting of μ_1, ν_1 and of μ_2, ν_2 respectively. Clearly, $R'_1 \models \Sigma$, $R'_2 \models \Sigma$ (since Σ only containd FD's), $\pi_X(R'_1) = \pi_X(R'_2)$, $t[X \cap Y] \in \pi_{X \cap Y}(R'_1) = \pi_{X \cap Y}(R'_2)$, $T_u[R'_2] \models \Sigma$ and $T_u[R'_1]$ violates $Z \rightarrow A$.

Thus, Y is not a good complement of X iff there are two relations R'_1, R'_2 with at most two tuples each which witness this fact.

From the above observation, we can easily see that we can test if Y is a good complement of X by doing the following for each FD $Z \rightarrow A$ in Σ:

Initialize T_1 to be a relation with three tuples μ_1, ν_1, t_1 as follows:

$t_1[W]=a_0$ for each W in U,

$\nu_1[W]=a_0$ for W in Y, a_1 for W in X-Y,

$\mu_1[W]=a_0$ for W in Z, a_2 for W in U-Z.

Initialize T_2 to be a relation with three tuples μ_2, ν_2, t_2 as follows:

$t_2=t_1$, $\nu_2=\nu_1$, $\mu_2[W]=a_2$ for each W in U.

Repeat until no new change is made on either T_1 or T_2:

Compute the chase wrt Σ of μ_1, ν_1 (in this and all subsequent chases, to equate a_i and a_j, $i < j$, replace a_j by a_i).

$\mu_2[X] \leftarrow \mu_1[X]$, $\nu_2[X] \leftarrow \nu_1[X]$.

Compute the chase wrt Σ of μ_2, ν_2, of ν_2, t_2, and of μ_2, t_2.

$\mu_1[X] \leftarrow \mu_2[X]$, $\nu_1[X] \leftarrow \nu_2[X]$.

When the above procedure terminates, we check if $\mu_1[A]=t_1[A]$. If not, then T_1, T_2 constitute a *counterexample* to the goodness of Y; if it turns out that $\mu_1[A]=t_1[A]$ for each *FD* $Z \rightarrow A$ in Σ, then we have actually *proved* that no such counterexample with at most two tuples in each relation can exist, and so Y is a good complement of X.

Since each execution of the repeat loop can be done in time $O(|\Sigma|)$, and each time we lose at least one out of $O(|U|)$ symbols, the running time of the algorithm is $O(|\Sigma|^2\ |U|)$ (the procedure is repeated at most $|\Sigma|$ times).

Notice that, if Y happens to be a good complement of X, then actually Test 2 accepts precisely the translatable insertions, whereas in the opposite case it rejects all insertions. However, testing if Y is a good complement of X can be done once and for all at the time Y is declared as the complement to use, and if it is found to fail then the database system can simply disregard Test 2.

3.2. Complexity of Testing Translatability

So far, we have shown how one can test if a proposed insertion of a tuple into a view is translatable, and if so, how to do the translation (Theorem 3). We presented an $O(|V|^3 log|V|)$ algorithm for testing translatability. Since this algorithm is likely to be inefficient in practice, we also developed two alternative stronger tests, which can be executed faster.

In the sequel, we are going to prove a result which has some negative implications regarding the extent to which one can hope to improve the running time required to test translatability. Specifically, we will show that, if the view is presented in an *exponentially succinct* way (i.e. as a union of Cartesian products) then testing translatability becomes Π_2^p*-hard* [St]. This result provides strong evidence against the possibility of having an algorithm that runs in time less than $O(|V|)$, i.e. it indicates that the whole view has to be examined in order to test translatability.

Moreover, we believe that this result also casts some doubt on the possibility of substantially improving the running time of our algorithm. Loosely speaking, Π_2^p-hardness seems to indicate that the problem lacks a "nice" combinatorial structure, which could be exploited to yield an algorithm considerably more efficient than the one resulting from our (more or less) straightforward approach.

We will now prove the result:

Theorem 4:

Determining if an insertion is translatable is Π_2^p-hard if the view V is given implicitly as the union of two Cartesian products, of total size $O(|U|)$.

Proof: Let G be a Boolean formula in 3-CNF, containing the variables x_i, $i=1,...,n$, and consisting of clauses f_j, $j=1,...,m$, and let $X=\{x_1, \dots, x_k\}$, $Y=\{x_{k+1}, \dots, x_n\}$ be a given partition of the set of variables of G. It is known [St, W] that determining if for all possible assignments of truth values to the variables in X G is satisfiable, i.e. if $\forall X \exists Y\ G(X, Y)=1$ (where $\forall X$ means $\forall x_1 ... \forall x_k$ etc.) is Π_2^p-complete. In what follows we give a polynomial-time

reduction from this problem to the problem of testing translatability of an insertion to a succinctly presented view.

Let U be $BX_1X'_1...X_nX'_nAF_1...F_mC$, and let Σ consist of the FD's $X_1X'_1...X_kX'_k \rightarrow A$, $F_1...F_m \rightarrow C$, $BA \rightarrow C$, and, for each clause $f_j = l_{j1} + l_{j2} + l_{j3}$ of G, the FD's $L_{j1}A \rightarrow F_j$, $L_{j2}A \rightarrow F_j$, $L_{j3}A \rightarrow F_j$ (where L_{ji} is X_r if l_{ji} is x_r, and L_{ji} is X'_r if l_{ji} is $\neg x_r$). Let the view be $BX_1X'_1...X_nX'_n$, and let the complementary view be $X_1X'_1...X_nX'_nAF_1...F_mC$. Finally, let the instance V of the view be $s_B \times s_{X_1X'_1} \times ... \times s_{X_nX'_n} \cup s$, where $s_{X_iX'_i}$ is a relation over $X_iX'_i$ consisting of two tuples μ_i, ν_i with $\mu_i[X_i]=0$, $\mu_i[X'_i]=1$, $\nu_i[X_i]=1$, $\nu_i[X'_i]=0$, s_B is a single tuple over B with $s_B[B]=b$, and s is a single tuple over $BX_1X'_1...X_nX'_n$ with $s[B]=a$, $s[X_i]=1$, $s[X'_i]=1$. Observe that V is essentially just a list of all possible truth assignments: each tuple μ of V, with the exception of s, defines an assignment h: $\{x_1, \ldots, x_n\} \rightarrow \{0, 1\}$ by taking $h(x_i)=\mu[X_i]$ $(\mu[X'_i]=\neg\mu[X_i])$; also, $\mu[B]=b$.

Suppose now we want to insert in V the tuple t, where $t[B]=b$, $t[X_1X'_1...X_nX'_n]=s[X_1X'_1...X_nX'_n]$. We will show that this insertion is translatable iff $\forall X \exists Y\ G(X, Y)=1$. First, it is obvious that conditions (a) and (b) of Theorem 3 are satisfied. Furthermore, observing that the only tuple agreeing with t on $X_1X'_1...X_nX'_n$ (the common part of the view and the complement) is s, it is easy to see that condition (c) is satisfied if the FD f is taken to be $X_1X'_1...X_kX'_k \rightarrow A$ (because the only tuple agreeing with t on $X_1X'_1...X_kX'_k$ is s), or if f is $F_1...F_m \rightarrow C$ (since no attribute of f is in the view), or if f is $L_{ji}A \rightarrow F_j$ (since s agrees with t on all possible L_{ji}'s).

Thus, all we have to show is that, for all tuples r with $r \neq s$ (these are the tuples agreeing with t on B), $Chase_{\Sigma}[R(V, t, r, BA \rightarrow C)]$ succeeds (i.e. starting with $r[A]=s[A]$ we eventually equate $r[C]$, $s[C]$) iff there is a satisfying assignment h for G which agrees with the one defined by r on $\{x_1, \ldots, x_k\}$.

First, suppose there is such an assignment h, and let r_h be the tuple corresponding to it. Since $r_h[X_1X'_1...X_kX'_k]=r[X_1X'_1...X_kX'_k]$, $r_h[A]=r[A]$, so $r_h[A]=s[A]$. Since h satisfies f_j, $r_h[L_{ji}]=1$ for some i, so $r_h[L_{ji}]=s[L_{ji}]$, i.e. $r_h[L_{ji}A]=s[L_{ji}A]$, and so $r_h[F_j]=s[F_j]$, for $j=1,\ldots,m$. Thus, $r_h[F_1...F_m]=s[F_1...F_m]$, so $r_h[C]=s[C]$. But since $r_h[BA]=r[BA]$, $r_h[C]=r[C]$, and thus

$r[C]=s[C]$, i.e. $\text{Chase}_\Sigma[R(V,\ t,\ r,\ BA \rightarrow C)]$ succeeds. Conversely, it is not difficult to see (by essentially tracing the previous argument backwards) that $r[C]$, $s[C]$ can only be equated if there is a tuple corresponding to a satisfying assignment and agreeing with r on $X_1X'_1...X_kX'_k$.

Thus, we have established that the insertion of t into V is translatable iff $\forall X \exists Y\ G(X,\ Y)=1$. Since U, Σ, t and the description of V as a Cartesian product can obviously be constructed from G, X, Y in polynomial time, we are done. ∎

It certainly is not surprising that using a similar (only simpler) construction we can show an analogous result for Test 1:

Theorem 5:

Determining if Test 1 accepts an insertion is co-*NP*-complete if the view V is given implicitly as the union of two Cartesian products, of total size $O(|U|)$.

Proof: We first show membership in co-*NP*: if X denotes the view and Y the complement, then the following is a non-deterministic polynomial time algorithm to determine if Test 1 *does not accept* the insertion of the tuple t into V: guess an *FD* $Z \rightarrow A$ in Σ and two tuples r, μ over X, and verify that $r \in V$, $\mu \in V$, $r[Z \cap X]=t[Z \cap X]$ (and $r[A] \neq t[A]$, if $A \in X$), $\mu[X \cap Y]=t[X \cap Y]$; construct a relation R consisting of two tuples r', μ', with $r'[X]=r$, $\mu'[X]=\mu$, and with new symbols in the Y-X columns, only with $r'[Z \cap (Y\text{-}X)]=\mu'[Z \cap (Y\text{-}X)]$; compute the chase wrt Σ of R, and verify that it does not attempt to equate two distinct elements of r, μ, and, if $A \in Y\text{-}X$, it does not equate $r[A]$, $\mu[A]$.

To prove the hardness part, we will make a reduction from unsatisfiability of Boolean formulas in 3-CNF. Let G be such a formula, with variables x_i, $i=1,...,n$, and clauses f_j, $j=1,...,m$. Let U be $BX_1X'_1...X_nX'_nC$, and let Σ consist of the *FD*'s $B \rightarrow C$, and, for each clause $f_j=l_{j1}+l_{j2}+l_{j3}$ of G, the *FD* $L_{j1}L_{j2}L_{j3} \rightarrow C$; let the view be $BX_1X'_1...X_nX'_n$, the complement be $X_1X'_1...X_nX'_nC$, and the instance V of the view be $s_B \times s_{X_1X'_1} \times ... \times s_{X_nX'_n} \cup s$, where s_B, $s_{X_iX'_i}$ are as in the Proof of Theorem 4, and $s[B]=a$, $s[X_i]=0$, $s[X'_i]=0$.

Suppose now we want to insert in V the tuple t, where $t[B]=b$,

$t[X_1X'_1...X_nX'_n]=s[X_1X'_1...X_nX'_n]$; we claim that this insertion is accepted by Test 1 iff G is unsatisfiable. To see this, observe that the only tuple of V agreeing with t on $X_1X'_1...X_nX'_n$ (the common part of the view and the complement) is s; by a reasoning similar to that in the Proof of Theorem 4, we see that the only *FD* which needs to be checked is $B \rightarrow C$. Now if $r \neq s$, the chase on r, s will equate $r[C]$, $s[C]$ iff for some j, $r[L_{j1}L_{j2}L_{j3}]=s[L_{j1}L_{j2}L_{j3}]$, i.e. $r[L_{ji}]=0$, which means that the assignment corresponding to r does not satisfy f_j. Since this should happen for all tuples r, the claim is established.

Since U, Σ, t and the description of V as a Cartesian product can be constructed from G in polynomial time, the proof is complete. ∎

3.3. Finding a Complement

So far, we have assumed the following scenario for translating view updates: when the user updates a view, he also specifies unambiguously the semantics of the update by defining a complement which should be kept constant during the translation. We studied in detail the problem of checking if a proposed insertion of a tuple into a projective view is translatable, when the complement is another projection and the database consists of a single relation satisfying a given set of functional dependencies.

However, a real database system should also be able to provide the user with some assistance concerning the task of defining a complement. We already gave a glimpse at this problem in Section 2, where, after we characterized complementary views, we examined the problems of finding a nonredundant complement and a minimum complement. Now that we have also gained some understanding of testing translatability, we can pose the following question: Suppose the user wishes to have the update translatable, imposing only partial (or none at all) restriction on the complement to be used. How can one determine a complement which will render the update translatable?

Let the view be X, and suppose Y is a complement of X such that the insertion of the tuple t into the instance V of the view is translatable under constant Y. Clearly, $Y = W \cup (U\text{-}X)$, where $W \subseteq X$. Since $t[W] \in \pi_W(V)$ (condition (a) of Theorem 3), there is a tuple r of V such that $r[W] = t[W]$. Consider now the set of attributes $W_r = \{A\colon A \in X,\ r[A] = t[A]\}$. It is immediate that $t[W_r] \in \pi_{W_r}(V)$, $\Sigma \models W_r \rightarrow Y$ (since $\Sigma \models W \rightarrow Y$ by condition (b) of Theorem 3, and $W_r \supseteq W$), and Σ does not imply $W_r \rightarrow X$ (if $\Sigma \models W_r \rightarrow X$, then the insertion of t into V is not translatable since $t[W_r] = r[W_r]$, $t \neq r$); moreover, if R is a database such that $R \models \Sigma$, $\pi_X(R) = V$, then $t^* \pi_{W_r}(R) = t^* \pi_W(R)$, and thus since $R \cup t^* \pi_W(R) \models \Sigma$, it follows that also $R \cup t^* \pi_{W_r}(R) \models \Sigma$, for all such R. Therefore, the insertion of t into V is also translatable under constant $Y_r = W_r \cup (U\text{-}X)$.

From the above discussion, it is easy to see the following:

Theorem 6:

Given Σ, X, V and t, we can find a complement Y of X such that the insertion of t into V is translatable under constant Y within $min(|V|, 2^{|X|})$ tests of translatability.

Proof: One can compute, for each tuple r of V, the set $W_r=\{A\colon A \in X, r[A]=t[A]\}$, and, after eliminating duplications, test, for each such W_r, if the insertion of t into V is translatable under constant $Y_r= W_r \cup (U-X)$. If no such W_r is found, then, by the preceding discussion, there is no complement Y of X such that the insertion of t into V is translatable under constant Y. ∎

Thus, we can determine if there is a complement which renders a given insertion translatable in polynomial time (see the Corollary to Theorem 3). Observe, however, that the polynomial complexity depends strongly on the fact that we are allowing the whole view V as part of the problem instance. The following result indicates that there is an inherent exponential dependence on $|U|+log|V|$; in other words, we may nevertheless have to check all possible subsets of X in order to find a complement.

Theorem 7:

Determining if there is a complement Y of X such that the insertion of t into V is translatable under constant Y is NP-hard if the view V is presented succinctly (as in Theorem 4).

Proof: Let G be a Boolean formula in 3-CNF, containing the variables x_i, $i=1,...,n$, and consisting of clauses f_j, $j=1,...,m$; assume furthermore with no loss of generality that the variables appearing in each clause are distinct. Let U be $X_1X'_1...X_nX'_nF_1...F_m$, and, for each clause $f_j=l_{j1}+l_{j2}+l_{j3}$ of G, let Σ contain the FD's $L_{j1} \rightarrow F_j$, $L_{j2} \rightarrow F_j$, $L_{j3} \rightarrow F_j$. Let the view X be $X_1X'_1...X_nX'_n$, and the instance V of the view be $s_{X_1X'_1} \times ... \times s_{X_nX'_n}$, where $s_{X_iX'_i}$ is as in the Proof of Theorem 4.

Suppose now we want to insert in V the tuple t, where $t[X_i]=t[X'_i]=1$. We will show that there is a complement $Y=W \cup F_1...F_m$ $(W \subseteq X)$ such that the insertion of t into V is translatable under constant Y, iff G is satisfiable. First, it is easy to see that, since $t[W]$ should be in $\pi_W(V)$, W should contain at most one of the attributes X_i, X'_i, for each i; furthermore, since we should

have that $\Sigma \models W \rightarrow F_1 ... F_m$ (clearly Σ does not imply $W \rightarrow X$), it is not difficult to see that W should define a satisfying assignment for G.

To complete the proof, observe that, if R is a database such that $R \models \Sigma$, $\pi_X(R) = V$, then for any two tuples μ, ν of R, $\mu[F_j] = \nu[F_j]$. This happens because, if $\mu[L_{j1}] \neq \nu[L_{j1}]$ and $\mu[L_{j2}] \neq \nu[L_{j2}]$ (in the opposite case obviously $\mu[F_j] = \nu[F_j]$), then there is a tuple ξ in R such that $\xi[L_{j1}] = \mu[L_{j1}]$, $\xi[L_{j2}] = \nu[L_{j2}]$ (because the variables in f_j are distinct); thus, $\xi[F_j] = \mu[F_j]$, $\xi[F_j] = \nu[F_j]$, and therefore $\mu[F_j] = \nu[F_j]$. It follows that, for the insertion of t into V to be translatable under constant Y, it suffices to have $t[W] \in \pi_W(V)$ and $\Sigma \models W \rightarrow F_1 ... F_m$.

Finally, U, Σ, X, t and the description of V as a Cartesian product can be constructed from G in polynomial time, and we are done. ∎

We remark that, by following a similar line of reasoning, one can see that Theorem 6 remains true if we interpret "translatable" as "accepted by Test 1 (Test 2)", and "test of translatability" as "Test 1 (Test 2)". The same holds for Theorem 7 (the reductions, although somewhat subtler, are based on the same idea as the reduction given in the Proof of Theorem 7, and are therefore omitted).

4. The Translation of Deletions and Replacements

In this Section we briefly show how the ideas developed previously for the case of translating the insertion of a tuple to a view can be adapted in a straightforward manner to handle the case of deleting a tuple and of replacing a tuple with another. We continue to assume that Σ is a set of *FD*'s satisfied by the database R, and that we are given the view X, the complement Y and the current instance V of the view.

4.1. Deletions

Suppose we wish to translate the update u on the view consisting of the *deletion* of a tuple t, $t \in V$, while keeping the complement $\pi_Y(R)$ constant. The update T_u on R which achieves this should satisfy $\pi_X(T_u[R]) = V - t$, $\pi_Y(T_u[R]) = \pi_Y(R)$, and also $T_u[R] \models \Sigma$ for all R such that $R \models \Sigma$, $\pi_X(R) = V$ (compare with Properties A through D given for the case of an insertion).

Now since $\pi_Y(R)$ must be kept constant, we must have that $t[X \cap Y] \in \pi_{X \cap Y}(V-t)$; in other words, there is a tuple $r \in V$ such that $r \neq t$, $r[X \cap Y] = t[X \cap Y]$. From this we now see that $X \cap Y$ cannot be a superkey of X (since V is a projection of a legal instance), so by Theorem 1, $X \cap Y \rightarrow Y$. It follows that the only possible candidate for T_u is the deletion of the *tuple* $t * \pi_Y(R)$ from the database R: $T_u[R] = R - t * \pi_Y(R)$.

But now observe that, since $T_u[R] \subseteq R$ and Σ only containds *FD*'s, $T_u[R] \models \Sigma$ if $R \models \Sigma$. Thus, our last requirement that $T_u[R] \models \Sigma$ for all R such that $R \models \Sigma$, $\pi_X(R) = V$, is satisfied trivially.

We have thus shown the following:

Theorem 8:

The deletion of t from V is translatable as $R \leftarrow R\text{-}t^{*}\pi_{Y}(R)$ if and only if

(a) $t[X\cap Y]\in\pi_{X\cap Y}(V\text{-}t)$.

(b) Σ implies $X\cap Y\rightarrow Y$, and Σ does not imply $X\cap Y\rightarrow X$. ∎

Hence, determining if a deletion is translatable can be done in time $O(|V|+|\Sigma|)$.

4.2. Replacements

Suppose now the update we wish to translate under constant complement Y is the *replacement* of a tuple t_1, $t_1\in V$, by a tuple t_2, $t_2\notin V$. The update T_u on R should satisfy $\pi_X(T_u[R])=V\text{-}t_1 \cup t_2$, and again $\pi_Y(T_u[R])=\pi_Y(R)$, and $T_u[R]\models\Sigma$ for all R such that $R\models\Sigma$, $\pi_X(R)=V$. We distinguish two cases:

Case 1 $t_1[X\cap Y]\neq t_2[X\cap Y]$.

This case exhibits a behavior similar to the one we are already familiar with: specifically, since $\pi_Y(R)$ must be kept constant, we must have $t_1[X\cap Y]\in\pi_{X\cap Y}(V\text{-}t_1)$, $t_2[X\cap Y]\in\pi_{X\cap Y}(V)$. From this it follows that $X\cap Y$ cannot be a superkey of X, and thus it is a superkey of Y by Theorem 1. Hence, the only possible candidate for T_u is the replacement of the *tuple* $t_1{}^{*}\pi_Y(R)$ by the *tuple* $t_2{}^{*}\pi_Y(R)$.

To check now if the last condition is satisfied, i.e. if $T_u[R]\models\Sigma$ for all R such that $R\models\Sigma$, $\pi_X(R)=V$, it is not difficult to see (by a reasoning exactly analogous to the one given for insertion) that all we have to do is check if $\text{Chase}_{\Sigma}[R(V, t_2, r, f)]$ succeeds for all *FD*'s f in Σ and for all tuples r in V *which are different from* t_1.

Case 2 $t_1[X\cap Y]=t_2[X\cap Y]$.

In this case we see that the first two conditions can be satisfied with no further restrictions on V, t, X, or Y, and moreover the only possible candidate for T_u is replacing the *set of tuples* $t_1{}^{*}\pi_Y(R)$ by the *set of tuples* $t_2{}^{*}\pi_Y(R)$ (we can no longer assert as before that either set will

consist of a single tuple, since this depended on $X \cap Y$ being a superkey of Y, which is no longer necessary).

Checking whether the last condition is satisfied, i.e. whether $T_u[R] \models \Sigma$ for all R such that $R \models \Sigma$, $\pi_X(R) = V$, can still be done by checking if $\text{Chase}_\Sigma[R(V, t_2, r, f)]$ succeeds for all f in Σ and for all r in V, $r \neq t_1$ (one can see that the fact that $t_1 * \pi_Y(R)$ and $t_2 * \pi_Y(R)$ may consist of more than one tuple does not affect anything).

Thus, we have the following:

Theorem 9:

The replacement of t_1 by t_2 in V $(t_1 \in V,\ t_2 \notin V)$ is translatable as $R \leftarrow R - t_1 * \pi_Y(R) \cup t_2 * \pi_Y(R)$ if and only if

(a) $t_1[X \cap Y] \in \pi_{X \cap Y}(V - t_1)$ and $t_2[X \cap Y] \in \pi_{X \cap Y}(V_2)$, or $t_1[X \cap Y] = t_2[X \cap Y]$.

(b) Σ implies $X \cap Y \rightarrow Y$ and Σ does not imply $X \cap Y \rightarrow X$, or $t_1[X \cap Y] = t_2[X \cap Y]$.

(c) $\text{Chase}_\Sigma[R(V, t_2, r, f)]$ succeeds for all f in Σ and for all r in V, $r \neq t_1$. ∎

From Theorem 9 it should be clear that one can develop results analogous to the ones given for the case of insertion in a straightforward way. Thus, we will not pursue this direction any further.

5. Explicit Functional Dependencies

Functional dependencies assert that a certain mapping is one-to-one -for example, a mapping from employee-project pairs to managers, or from cost-price pairs to rates of profit. However, there is a difference; certain such mappings are essential information stored by the database (as in the first example above), whereas others are *redundant information*, mappings that could be computed explicitly (as in the second example). We call the latter case of *FD*'s *explicit FD*'s (*EFD*'s).

EFD's are important in the context of views and view complements, because they can seriously affect the information content of database mappings. We thus felt that we should study their behavior vis-a-vis the other known classes of dependencies. We first define formally what an *EFD* is:

Definition:

A set of attributes X *explicitly determines* a set of attributes Y (notation: $X \rightarrow_e Y$) if there is an *instance-independent* function f (called a *witness* of $X \rightarrow_e Y$) such that $\pi_{XY}(R)=f(\pi_X(R))$, for any legal instance R of the database.

Examples. Cost-Profitrate$\rightarrow_e$ Price, Course-Student-Grade$\rightarrow_e$ Average-Grade.

We remark that, in our definition of an *EFD*, no special property of the witness function f is assumed. This leads naturally to the following extension of the meaning of *implication* of an *EFD* σ from a set of dependencies Σ, where σ_i, $i=1,...,k$, are the *EFD*'s in Σ: *for all functions* f_i, $i=1,...,k$, *there is a function* f such that, if a database R satisfies all dependencies in Σ (where f_i is taken as the witness of σ_i), then it also satisfies σ (where f is taken as the witness of σ). In case σ is not an *EFD*, then one just omits the requirement of the existence of f.

As we are going to see shortly, with this approach *EFD*'s behave very much like *FD*'s (in the sense of Propositions 1 and 2). It would be interesting to see what happens if one imposes natural restrictions on the witness function f, such as invertibility, 0-1-valuedness, etc.

In the following, if Σ is a set of dependencies, we denote by Σ_F the set of *FD*'s $\{X \rightarrow Y:\ X \rightarrow_e Y \text{ is in } \Sigma\}$.

Proposition 1:

Let Σ be a set of *EFD*'s; then $\Sigma \models X \rightarrow_e Y$ iff $\Sigma_F \models X \rightarrow Y$.

Proof: Consider the following chase procedure for computing X^+: initialize X^+ to X; repeatedly locate a member $Z \rightarrow B$ of Σ_F such that $Z \subseteq X^+$ and B is not contained in X^+, and set X^+ to $X^+ \cup B$. As is well known [MMS], this procedure terminates with a unique X^+, and furthermore $\Sigma_F \models X \rightarrow Y$ iff $Y \subseteq X^+$.

We will now argue that also $\Sigma \models X \rightarrow_e Y$ iff $Y \subseteq X^+$. First, if $Y \subseteq X^+$, then it is clear that $\Sigma \models X \rightarrow_e Y$ by the construction of X^+ (observe that, if $X \rightarrow_e Y$ and $Y \rightarrow_e Z$, then $X \rightarrow_e Z$). Conversely, if Y is not contained in X^+, we will show that Σ does not imply $X \rightarrow_e Y$. For each *EFD* $Z \rightarrow_e B$ in Σ, pick as its witness a function $f_{Z \rightarrow B}$ such that $f_{Z \rightarrow B}(t_Z) = t_{ZB}$, where t_Z is a tuple over Z with $t_Z[W] = a$ for all W, and t_{ZB} is a tuple over ZB with $t_{ZB}[W] = a$ for all W. Now if g is a purported witness of $X \rightarrow_e Y$, then consider the database R consisting of a single tuple t with $t[W] = a$ for $W \in X^+$, and $t[W] = y$ otherwise, where $y \neq \pi_A(g(t[X]))$, for some A in Y-X. It is clear that R satisfies each *EFD* $Z \rightarrow_e B$ in Σ (with witness $f_{Z \rightarrow B}$), but R does not satisfy $X \rightarrow_e Y$ with witness g. ∎

Proposition 2:

Let Σ be a set of *EFD*'s, and let Σ' be a set of *FD*'s and *JD*'s. Suppose that $\Sigma \cup \Sigma' \models \sigma$:

(a) If σ is an *FD* or *JD* or *embedded JD*, then $\Sigma_F \cup \Sigma' \models \sigma$.

(b) If σ is an *EFD*, then $\Sigma \models \sigma$.

Proof:

(a) If $\Sigma_F \cup \Sigma'$ does not imply σ, then there is a relation R which satisfies $\Sigma_F \cup \Sigma'$ but violates σ. Now since $R \models \Sigma_F$, clearly we can pick a function f_i for each *EFD* σ_i in Σ such that R also satisfies σ_i with witness f_i. Thus, R satisfies $\Sigma \cup \Sigma'$, and therefore $\Sigma \cup \Sigma'$ does not imply σ.

(b) Assume that Σ does not imply σ, and observe that the one-tuple relation R constructed in

the Proof of Proposition 1 also satisfies Σ' (since it satisfies any *FD*, *JD* or embedded *JD*. Thus, R satisfies $\Sigma \cup \Sigma'$ and violates σ, and so $\Sigma \cup \Sigma'$ does not imply σ. ∎

Thus, we can easily augment any of the known axiom systems for *FD*'s, *FD*'s and *MVD*'s etc. to include *EFD*'s. Moreover, our characterization of complementary views (Theorem 1) can be extended to include *EFD*'s as follows:

Theorem 10:

Let Σ be a set of *FD*'s, *JD*'s and *EFD*'s. Then X, Y are complementary iff:

(a) They are complementary when considered as views of $\pi_{X \cup Y}(R)$ (i.e., Σ implies the embedded *MVD* $X \cap Y \to\to X\text{-}Y | Y\text{-}X$); and

(b) $\Sigma \models X \cup Y \to_e U$.

Proof: The "if" direction is immediate: from (a) $\pi_X(R)*\pi_Y(R) = \pi_{X \cup Y}(R)$ for every legal database R, and then from (b) $R = f(\pi_{X \cup Y}(R)) = f(\pi_X(R)*\pi_Y(R))$, where f is an instance-independent function. Thus, if for two legal instances R, R' we have $\pi_X(R) = \pi_X(R')$ and $\pi_Y(R) = \pi_Y(R')$, we get $R = f(\pi_X(R)*\pi_Y(R)) = f(\pi_X(R')*\pi_Y(R')) = R'$, i.e. X, Y are complementary.

For the "only if" direction, assume first that (a) is false, i.e. Σ does not imply the embedded *MVD* $X \cap Y \to\to X\text{-}Y | Y\text{-}X$. We first remark that the Equivalence Theorem of [SDSF] is also true if σ is an embedded *MVD* (using the partial extension of the equivalence between dependencies and formulas to include embedded *MVD*'s described in Section 7; the 2-tuple Subrelation Lemma can be extended to the case in which σ is an embedded *MVD*, by an argument analogous to the one given for the case in which σ is an *MVD*). Using the same construction as in the Proof of Theorem 1 (combined with Proposition 2 (a) and the above observation), we obtain two distinct two-tuple relations R, R' such that $\pi_X(R) = \pi_X(R')$, $\pi_Y(R) = \pi_Y(R')$, and R, R' satisfy all the *FD*'s and *JD*'s in Σ and all the *FD*'s in Σ_F. Then it is easy to see that we can pick, for each *EFD* σ in Σ, a function f such that both R and R' satisfy σ with witness f. This shows that X, Y are not complementary.

If (b) is false, then $(X \cup Y)^+ \neq U$, where $(X \cup Y)^+$ is the closure of $X \cup Y$ wrt Σ_F. Let R, R' be two one-tuple relations such that $R[W]=R'[W]=a$ for W in $(X \cup Y)^+$, and $R[W] \neq R'[W]$ otherwise. Clearly, $R \neq R'$, $\pi_X(R)=\pi_X(R')$, $\pi_Y(R)=\pi_Y(R')$, R, R' satisfy all *FD*'s and *JD*'s in Σ, and moreover by picking as the witness of an *EFD* $Z \rightarrow B$ in Σ a function $f_{Z \rightarrow B}$ as in the Proof of Proposition 1, we see that R, R' also satisfy the *EFD*'s in Σ. This shows that X, Y are not complementary, and the proof is complete. ∎

Intuitively, Theorem 1 stated that, if the only dependencies present are *FD*'s and *JD*'s, then the only way to reconstruct a database from two projections is by join. Theorem 6 states that, if *EFD*'s are also present, then the only way is to join the two projections and then ***explicitly compute*** the information which is still missing.

6. Conclusions and Directions for Further Research

In this work, we have studied some of the computational problems arising when one considers applying, in the context of the relational model, the methodology suggested by Bancilhon and Spyratos for translating view updates. We discovered that certain important problems such as testing translatability and determining a complement which renders an update translatable, although solvable in polynomial time (Theorems 3, 6, 8, 9), exhibit an interesting kind of inherent complexity (Theorems 4, 5, 7), which indicates the existence of limitations on how efficiently they can be solved. However, we have only concentrated on a very simple case of the application; we feel that much remains to be done before a reasonable account of the applicability of the methodology can be attempted. In particular, the following possibilities seem to us to be worthy of further investigation:

(1) Allowing more general dependencies: In particular, it would be interesting to see to what extent can Theorem 1 be generalized, especially in view of the negative result of [V1]. More importantly, though, one should study the problem of testing translatability and designing a translation (recall that we found the translation of deletions to be trivial just because we only considered functional dependencies). It is conceivable that our basic idea of a chase-type algorithm will be useful, although we cannot see to what extent.

(2) Considering views that are a *restriction* of a projection (i.e. of the form $\sigma_P \pi_X$, where P is a predicate on tuples): It should be noted that most of the views occuring in practice are actually of the above form. The complement here can be a *pair of views*, e.g. $(\sigma_{\neg P}, \sigma_P \pi_Y)$ or $(\sigma_{\neg P} \pi_X, \pi_Y)$, where π_Y is a complement of π_X. We believe that, in the case of only functional dependencies (which is still very important from a practical viewpoint), our basic approach can be used with only simple modifications (at least for certain P's).

(3) Considering multi-relation databases with views that are projections of joins of relations: this is most important, given that the universal relation assumption is being criticized as

unrealistic. We also believe that this is likely to be the theoretically most interesting direction.

(4) Studying the explicit functional dependencies: It seems to us that *EFD*'s are a step in the right direction, if one wants a model capable of capturing the information content of database mappings. We have already examined their influence on complementarity of views (Theorem 10). Their effect on issues like testing translatability or designing a translation (perhaps in conjunction with refining our definition to capture more semantics) is a question which we feel deserves further research.

References

[Ar] Armstrong, W.W. Dependency structures of database relationships. Proc. IFIP 74, North Holland, Amsterdam, 1974, pp. 580-583.

[As] Astrahan, M.M., et al. System R: Relational approach to database management. ACM Transactions on Database Systems 1, 2 (June 1976), 97-137.

[BB] Beeri, C., and Bernstein, P.A. Computational problems related to the design of normal form relational schemas. ACM Transactions on Database Systems 4, 1 (March 1979), 30-59.

[BBG] Beeri, C., Bernstein, P.A., and Goodman, N. A sophisticate's introduction to database normalization theory. In Proc. 4th VLDB Conference, West Berlin, September 1978.

[BS] Bancilhon, F., and Spyratos, N. Update semantics of relational views. ACM Transactions on Database Systems 6, 4 (December 1981), 557-575.

[BV] Beeri, C., and Vardi, M.Y. On the complexity of testing implications of data dependencies. Research Report, Department of Computer Science, The Hebrew University of Jerusalem, Jerusalem, Israel, December 1980.

[Ca] Carlson, C.R., and Arora, A.K. The updatability of relational views based on functional dependencies. Third International Computer Software and Applications Conference, IEEE Computer Society, Chicago, IL, November 1979.

[Ch] Chamberlin, D.D., et al. Views, authorization and locking in a relational data base system. In Proc. 1975 Nat. Computer Conf., AFIPS Press, Arlington, Va.

[Co1] Codd, E.F. A relational model for large shared data banks. Communications of the ACM 13, 6 (June 1970) 377-387.

[Co2] Codd, E.F. Relational completeness of database sublanguages. In Data Base Systems,

R.Rustin, Ed., Prentice Hall, Englewood Cliffs, N.J., 1972, pp. 65-98.

[Co3] Codd, E.F. Further normalization of the database relational model. In Database Systems, R.Rustin, Ed., Prentice Hall, Englewood Cliffs, N.J., 1972, pp. 33-64.

[Co4] Codd, E.F. Extending the database relational model to capture more meaning. ACM Transactions on Database Systems 4, 4 (December 1979), 397-434.

[Ck] Cook, S.A. The complexity of theorem proving procedures. Proceedings of the 3rd Annual ACM Symposium on the Theory of Computing. Shaker Heights, Ohio, May 1971, pp. 151-158.

[CO] CODASYL Data Base Task Group April 71 Report, ACM, New York.

[D] Date, C.J. An Introduction to Database Systems. Addison Wesley, Reading, Mass., 1977.

[DB] Dayal, U., and Bernstein, P.A. On the updatability of relational views. In Proc. 4th VLDB Conference, West Berlin, September 1978.

[F] Fagin, R. Multivalued dependencies and a new normal form for relational databases. ACM Transactions on Database Systems 2, 3 (September 1977), 262-278.

[FSD] Furtado, A.L., Sevcik, K.C., and Dos Santos, C.S. Permitting Updates through views of data bases. Information Systems 4, 4 (1979), 269-283.

[GJ] Garey, M.R. and Johnson, D.S. Computers and Intractability: A Guide to the Theory of NP-completeness (Freeman, San Francisco, CA, 1979).

[I] IMS/VS publications, IBM, White Plains, New York.

[K] Karp, R.M. Reducibility among combinatorial problems. In Complexity of Computer Computations, R.E. Miller and J.W. Thatcher, Eds., Plenum Press, New York, 1972, pp. 85-104.

[MMS] Maier, D., Mendelzon, A.O., and Sagiv, Y. Testing implications of data dependencies.

ACM Transactions on Database Systems 4, 4 (December 1979), 455-469.

[MSY] Maier, D., Sagiv, Y., and Yannakakis, M. On the complexity of testing implications of functional and join dependencies. JACM 28, 4 (October 1981), 680-695.

[R1] Rissanen, J. Independent components of relations. ACM Transactions on Database Systems 2, 4 (December 1977), 317-325.

[R2] Rissanen, J. Theory of relations for databases - a tutorial survey. Proc. 7th Symp. on Mathematical Foundations of Computer Science, Lecture Notes in Computer Science 64, Springer-Verlag, Berlin, Heidelberg, 1978, pp. 536-551.

[RS] Rowe, L., Stonebraker, M. Manuscript, U.C. Berkeley, 1979.

[Sp] Spyratos, N. Translation structures of relational views. In Proc. 6th VLDB Conference, Montreal, 1980.

[St] Stockmeyer, L.J. The polynomial time hierarchy. Theoretical Computer Science 3, 1 (1976), 1-22.

[SDSF] Sagiv, Y., Delobel, C., Stott Parker Jr., D., and Fagin, R. An equivalence between relational database dependencies and a fragment of propositional logic. Journal of the ACM 28, 3 (July 1981), 435-453.

[SWKH] Stonebraker, M., Wong, E., Kreps, P., and Held, G. The design and implementation of INGRES. ACM Transactions on Database Systems 1, 3 (September 1976), 189-222.

[T] Todd, S.J.P. The Peterlee relational test vehicle - a system overview. IBM Systems Journal 15, 4, 285-308.

[U1] Ullman, J.D. Principles of Database Systems. Computer Science Press, Rockville, Maryland, 1980.

[U2] Ullman, J.D. The U.R. strikes back. Proc. of the ACM Symp. on Principles of Database Systems, Los Angeles, California, 1982.

[V1] Vardi, M.Y. On decomposition of relational databases. Proceedings of the 23rd Symposium on Foundations of Computer Science, Chicago, Illinois, 1982.

[V2] Vardi, M.Y. Inferring multivalued dependencies from functional and join dependencies. Research Report, Dep. of Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel, March 1980.

[W] Wrathall, C. Complete sets and the polynomial-time hierarchy. Theoretical Computer Science 3, 1, (1976), 23-33.

[Za1] Zaniolo, C. Analysis and design of relational schemata for database systems. Tech. Rep. UCLA-ENG-7669, Dep. of Computer Science, University of California, Los Angeles, California, July 1976.

[Za2] Zaniolo, C. Database relations with null values. Proc. of the ACM Symp. on Principles of Database Systems, Los Angeles, California, 1982.

[Zl] Zloof, M.M. Query-by-Example: a data base language. IBM Systems Journal 16, 4, 324-343.